# Dijkstra’s Algorithm

For this assignment, you are asked to write a java program that can read in a text file that contains multiple weighted-directed graphs and find the shortest path using Dijkstra’s algorithm for each graph.

**Input:** The input file is formatted as follows,

20 graphs in wdGraphs.txt.

\*\* G1: |V|=5, V={0,1,...4}

) E = {

|  |
| --- |
| (u, v, weight  ( 0, 2, |
| ( 1, 4, |
| ( 2, 0, |
| ( 2, 1, |
| ( 3, 0, |

10.869)

25.297)

8.403)

12.700)

89.156)

( 4, 3, 38.522)}

......

......

\*\* G20: |V|=455, V={0,1,...454}

(u, v, weight) E = {

...... }

The first line indicates the number of graphs in the file. There are 20 graphs in wdGraphs.txt. For each graph, V is the index set of vertices, i.e., V = {0,1,...n-1} where |V| = *n* is given. Each edge from *v* to *u* with weight *w* is presented as (*v, u, w*).

**Output:** The output shows the shortest path from *v*0 to *vn−*1 with incremental weight from *v*0 to each vertex on the path for each graph. The results should be shown on the screen formatted as follows.

Shortest Paths from vertex 0 to vertex n-1 in wdGraphs.txt, |V|=n G1’s shortest path from 0 to 4:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ( | | 0, | 2, | 10.869) | --> | 10.869 |
| ( | | 2, | 1, | 12.700) | --> | 23.569 |
| ( | | 1, | 4, | 25.297) | --> | 48.866 |
| G2’s | shortest 0 to 4:  ( 0, 3, 46.188) --> | | | | | 46.188 |
|  | ( 3, 4, 26.595) --> | | | | | 72.783 |

G3’s shortest path from 0 to 54:

\*\*\* There is no path.

# 

G20’s shortest path from 0 to 454:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ( 0, | 146, | 0.019) | --> | 0.019 |
| (146, | 117, | 10.144) | --> | 10.163 |
| (117, | 169, | 2.618) | --> | 12.781 |
| (169, | 454, | 43.630) | --> | 56.411 |

**Program requirement:** The name of the program should be dijkstra.java and I will compile and run your program on our Linux server as follows.

javac dijkstra.java

java dijkstra wdGraphs.txt

where wdGraphs.txt is the name of the input file. If your program fails to compile, you will get 0 point. I may test your program on a different graph file.

## There are two parts of submission: Programs (60%) and Reports (40%)

## Important!! You will lose significant points if you fail to follow the rules.

* Programs 60%, submission on Linux server. The score is based on the correctness and documentation of your programs.

1. Also, at the beginning of **every** method, class, or function that you developed on your own, put a few lines of comments with your name and date/time when you develop the code.
2. Do not declare any package for your programs. You can use any Java IDE on your local computer, but you have to transfer all programs and data files needed to our Linux server. Note that, some IDE may automatically add your programs/classes to some default package. You have to remove the package declaration after you transfer the program to a different location, otherwise, the grading script will fail to handle the special need and run into compilation problems.

**Report 40%.** You have to write up a report and prepare it in pdf format. The report should include the following items:

1. 10%. The program code is not required in the report. Instead, the *direct output* of your programs on the required input, which is graphs.txt, should be included immediately after the cover page. You can use Unix redirect command, >, to redirect your program’s output to a text file, then copy and paste to your report. Don’t fetch the screen shot. The output on the report must be consistent with the result I get when I run your program on graphs.txt.
2. 10%. A summary of the methods, algorithms and data structures, and the difficulties, if any, the project has faced and how to solve them.
3. 10%. Both time and space efficiency should be analyzed in terms of big-O notations.